On Scheduling Tasks With Exponential Service Times with In-Tree Precedence Contraints

John Bruno

June 1984

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS TR 84.3

RIACS

Research Institute for Advanced Computer Science

On Scheduling Tasks with Exponential Service Times with In-Tree Precedence Constraints

John Bruno

Research Institute in Advanced Computer Science
NASA Ames Reseach Center
and
The Department of Computer Science
University of California
Santa Barbara

ABSTRACT

In this paper we extend the work of Chandy and Reynolds in which they considered the problem of scheduling tasks on two identical processors. The processing times of the tasks are independent, identically distributed exponential random variables. The tasks are subject to in-tree precedence constraints. Chandy and Reynolds have shown that the expected value of the makespan(latest finishing-time) is minimized if and only if the scheduling policy is Highest-Level-First(HLF). We extend their result by showing that a policy maximizes the probability that all tasks finish by some time $t \geqslant 0$ if and only if the policy is HLF. Additionally, we show that a policy maximizes the probability that the sum of the finishing times of all the tasks is less than some value $s \geqslant 0$ if and only if the policy is HLF.

1. Introduction

In this paper we extend the work of Chandy and Reynolds[CR] in which they considered the problem of scheduling tasks on two identical processors. The processing times of the tasks are independent, identically distributed exponential random variables. The tasks are subject to in-tree precedence constraints. Chandy and Reynolds have shown that the expected value of the makespan(latest finishing-time) is minimized if and only if the scheduling policy is Highest-Level-First(HLF). We extend their result by showing that other criteria are also optimized by the HLF policy. In particular we show that a policy maximizes the probability that all tasks finish by some time $t \geqslant 0$ if and only if the policy is HLF. Additionally, we show that a policy maximizes the probability that the sum of the finishing times of all the tasks is less than some value $s \geqslant 0$ if and only if the policy is HLF.

In the next section we introduce definitions and notation used throughout the paper. Following this we give a precise formulation of the criteria for optimality and subsequently we state the main results. All proofs appear in the Appendix.

2. Definitions and Notation

A scheduling problem consists of a set of tasks, precedence constraints among the tasks, and service-time requirement, τ, for each task. In this paper we shall consider the service-time requirements of the tasks to be independent, identically distributed random variables with distribution function $F_\tau(t) = 1-e^{-\mu t}$. Below we introduce some graph-theoretic definitions which are useful for describing the precedence constraints among the tasks.

A *directed graph* G, is given by

1. V, a finite set of *tasks(vertices)*,
2. E, a finite set of *edges, and*
3. A *relation of incidence* which associates with each edge e in E an ordered pair of tasks (u,v). The tasks u and v called the *endpoints* of e and u (v) is called the *immediate predecessor(successor)* of v (u).

The notation $G(V,E)$ denotes the directed graph G with task set V and edge set E. $V(G)$ and $E(V)$ denote the task set and edge set of the directed graph G, respectively.

Let $G(V,E)$ be a directed graph. A task is called *initial(final)* if it has no immediate predecessors(successors). Let $I(G)$ denote the set of initial tasks of G. A *directed path* of *length* k is a sequence of $k \geqslant 1$ tasks $v_1, v_2, \ldots, v_k$ where v_i is an immediate predecessor of v_{i+1} for $i=1,..,k-1$. Task u (v) is called a *predecessor(successor)* of task v (u) if there is a directed path beginning at u and ending at v in G. We say that G is *acyclic* if there is no directed path in G with repeated tasks. If G is acyclic then we define $\lambda(v)$, the *level* of the task v, to be the length of the longest path from v to a final task. The graph formed from G by removing the task v belonging to G is denoted by G/v and consists of the tasks in $V-\{v\}$, the edges $e \in E$ such that both endpoints of e are in $V-\{v\}$, and the relation of incidence which is the restriction of the relation of incidence in G to the edges and tasks of G/v. G is called an *in-tree* if it is acyclic and each task has at most one immediate successor.

An instance of a *scheduling problem* is given by an in-tree G. As was mentioned above, each task of G has an associated service requirement τ. The service requirements of all tasks are independent, identically distributed random variables with distribution function $F_\tau(t) = 1-e^{-\mu t}$ where $\mu > 0$. We assume that there are 2 identical processors and the processing constraint is that a processor may not be assigned to a task until all of its predecessors have completed their processing.

A *policy*, π, is a mapping from in-trees into subsets of tasks satisfying:

1. $\pi(G)$ is a subset of $I(G)$, and
2. if $|I(G)|$[†] $\leqslant 2$ then $\pi(G) = I(G)$ otherwise $|\pi(G)| = 2$.

A policy is called *highest-level-first* (HLF) if $v \in \pi(G)$ then $\lambda(v) \geqslant \lambda(u)$ for all $u \in I(G)-\pi(G)$.

Given a scheduling problem G and a policy π, a *schedule* is determined by assigning the processors to the tasks determined by the policy π at *decision points*. The *decision points* consist of time zero and subsequent task completion times. At time zero the processors are assigned to the tasks in $\pi(G)$. The next decision point occurs at the first completion of a task in $\pi(G)$. If $\pi(G)$ contains two tasks the distribution function of the time to the first completion is $F_\tau(2t) = 1-e^{-2\mu t}$ and if $\pi(G)$ contains one task the distribution function of the completion time is $F_\tau(t)$. Suppose task $a \in \pi(G)$ is the first to finish. We now assign the processors to the tasks in $\pi(G/a)$ and the time to the next decision point is the time to the first completion of a task in $\pi(G/a)$. This process continues until all the tasks in G have been processed. Notice that preemptions are allowed.

3. Optimization Criteria

Let G be a scheduling problem, π a policy, and $t \geqslant 0$. We define $M_G^\pi(t)$ to be the *probability that all tasks in G complete their processing by time t given that scheduling begins at time zero and the policy π is used throughout.* We can write $M_G^\pi(t)$ as follows:

a) If G has no tasks (i.e. V and E are both empty) then

$$M_G^\pi(t) = 1.$$

[†] $|S|$ denotes the number of elements in the set S.

b) If G has exactly one initial task ($\pi(G) = \{a\}$) then

$$M_G^\pi(t) = \mu \int_0^t (1-F_\tau(x)) M_{G/a}^\pi(t-x)dx \ .$$

c) If G has two or more initial tasks ($\pi(G) = \{a, b\}$) then

$$M_G^\pi(t) = 2\mu \int_0^t (1-F_\tau(2x))(\frac{1}{2} M_{G/a}^\pi(t-x) + \frac{1}{2} M_{G/b}^\pi(t-x))dx \ .$$

Case a) is obvious. Case b) expresses $M_G^\pi(t)$ in terms of the probability that a finishes at time x and the probability of the remaining tasks finishing within time $t-x$. In a similar manner, case c) expresses $M_G^\pi(t)$ in terms of the probability that the first finish occurs at time x and the probability of the remaining tasks finishing within time $t-x$. There are two equally likely cases: either task a or task b finishes first.

We shall consider a second optimization criterion. Let G be a scheduling problem, π a policy, and $s \geqslant 0$. We define $W_G^\pi(s)$ to be the *probability that the sum of the finishing times of all tasks in G is no greater than s given that scheduling begins at time zero and the policy π is used throughout.* We can write $W_G^\pi(s)$ as follows:

a) If G has no tasks (i.e. V and E are both empty.) then

$$W_G^\pi(s) = 1 \ .$$

b) If G has exactly one initial task ($\pi(G) = \{a\}$) then

$$W_G^\pi(s) = \mu \int_0^{s/n} (1-F_\tau(x)) W_{G/a}^\pi(s-nx)dx \ ,$$

where $n = |V(G)|$.

c) If G has two or more initial tasks ($\pi(G) = \{a, b\}$) then

$$W_G^\pi(s) = 2\mu \int_0^{s/n} (1-F_\tau(2x))(\frac{1}{2} W_{G/a}^\pi(s-nx) + \frac{1}{2} W_{G/b}^\pi(s-nx))dx \ ,$$

where $n = |V(G)|$.

Case a) is obvious. Case b) expresses $W_G^\pi(s)$ in terms of the probability that a finishes at time x and the probability that the sum of the finishing times of the remaining tasks is not greater than $s-nx$. The idea is that by time x the finishing times of all n tasks are at least x and so we can reduce the original problem to the subproblem G/a with s reduced by nx. Case c) is similar to case b) except that there are two equally likely subcases to consider.

4. Optimality Equations

For $i = 1,2$ we define the operator L_i which maps the set of continuous functions on $[0,\infty)$ into itself. Let g be a continuous function of t on $[0,\infty)$.
Let

$$[L_i g](t) = i \mu \int_0^t (1-F_\tau(ix)) g(t-x)dx$$

Let G be an instance of a scheduling problem. Below we give a set of equations, called the *optimality equations*, which determine M_G, the optimal value for the makespan criterion. These equations are policy-independent and thus express the best we can hope to achieve under any circumstances. The optimality equations for M_G are:

a) If G is empty then

$$M_G = 1 \ .$$

b) If G has exactly one initial task, a, then

$$M_G = L_1 M_{G/a} .$$

c) If G has two or more initial tasks then

$$M_G = \frac{1}{2} \max_{\substack{a,b \in I(G) \\ a \neq b}} (L_2 M_{G/a} + L_2 M_{G/b}).$$

A policy π is called a *makespan optimal* policy if $M_G^\pi(t) = M_G(t)$ for all G and $t \geq 0$.

We next present the optimality equations for the sum-of-finishing-times criterion. For $i = 1,2$ and $n \geq 0$ we define the operator K_i^n which maps the set of continuous functions on $[0,\infty)$ into itself.

Let

$$[K_i^0 g] = g ,$$

and

$$[K_i^n g](s) = i \mu \int_0^{s/n} (1 - F_\tau(ix)) g(s - nx) dx$$

Below we give the *optimality equations* which determine W_G, the optimal value for the sum-of-finishing-times criterion. As before, these equations are policy-independent and thus express the best we can hope to achieve. The optimality equations for W_G are:

a) If G is empty then

$$W_G = 1 .$$

b) If G has exactly one initial task, a, and $|V(G)| = n+1$ then

$$W_G = K_1^n W_{G/a} .$$

c) If G has two or more initial tasks and $|V(G)| = n+1$ then

$$M_G = \frac{1}{2} \max_{\substack{a,b \in I(G) \\ a \neq b}} (K_2^n M_{G/a} + K_2^n M_{G/b}).$$

A policy π is called a *sum-of-finishing-times optimal* policy if $W_G^\pi(s) = W_G(s)$ for all G and $s \geq 0$.

5. Main Results

In this section we state the main results of this paper.

Theorem 1. A policy π is makespan optimal if and only if it is HLF.

Proof: See Appendix.

Theorem 2. A policy π is sum-of-finishing-time optimal if an only if it is HLF.

Proof: The proof of Theorem 2 follows the same pattern as the proof on Theorem 1 and is not given. See the Appendix for the proof of Theorem 1.

The pattern of the proof of Theorem 1 suggests that HLF is optimal with respect to a wide variety of criteria. Specifically, the proof uses the fact that the operator L_i is monotone. The inductive proof of Theorem A given in the Appendix will work for any monotone operator as long as certain special cases can be proved. The special cases consist of those where either H or G have exactly one initial task. In these cases we have used a sample-path analysis which works because of the simple structure of an in-tree with one initial task.

The policies we have defined are rather simple and do not depend on scheduling history. Because of the memoryless property of the exponential distribution, nothing is gained by going to a more complex class of policies.

It is known that HLF is not optimal for more than two processors[CR]. An out-tree is an acyclic directed graph in which each task has at most one immediate predecessor. For two or more machines, independent, identically distributed exponential service times and out-tree precedence constraints there are no known results. This is rather surprising, but true. It is tempting to try an consider these problems from a complexity point-of-view. The reader is referred to a recent paper by Christos Papadimitriou, which was inspired by an attempt to classify the complexity of these scheduling problems[Pa]. In contrast, the situation for the deterministic version of these problems reasonably well understood[Br].

6. References

[Br] Bruno, J. "Deterministic and Stochastic Scheduling Problems with Treelike Precedence Constraints," in *Deterministic and Stochastic Scheduling* eds. Dempster, M.A.H., et. al., D. Reidel Publishing Co., 1982, pp 367-374.

[CR] Chandy, K.M., and Reynolds, P.F., "Scheduling Partially Ordered Tasks with Exponentially Distributed Times," Department of Computer Sciences, University of Texas at Austin

[Pa] Papadimitriou, C. H., "Games Against Nature," *24th Symposium on the Foundations of Computer Science*, Nov 1983, pp 446-450

Appendix

In this appendix we give the proofs of the main results of this paper. These proofs are close to the ones given by Chandy and Reynolds[CR] for the case of expected makespan optimization. To begin with we define three relations over in-trees which were originally introduced in [CR].

Let $G(V,E)$ be an in-tree. Let N_i be equal to the number of tasks $v \in V$ such that $\lambda(v) = i$. Using N_i we define $S(G,i)$ as

$$S(G,i) = \sum_{k \geq i} N_k .$$

Let G and H be two in-trees. We define three relations, $\leq$, $\sim$, and $<$, on pairs of in-trees as follows:

1. If $S(G,H) \leq S(H,i)$ for all $i \geq 1$ then we write $G \leq H$.
2. If $S(G,H) = S(H,i)$ for all $i \geq 1$ then we write $G \sim H$.
3. If $G \leq H$ and there exists an i such that $S(G,i) < H(G,i)$ then we write $G < H$.

The above definitions compare in-trees in terms of the total number of nodes at and above each level. It turns out that the "flatter" the in-tree the better it is for maximizing M and W.

Lemma 1 Let G be an in-tree and $a,b \in I(G)$. If $\lambda(a) \geq \lambda(b)$ then $G/a \leq G/b$. If $\lambda(a) > \lambda(b)$ then $G/a < G/b$. If $\lambda(a) = \lambda(b)$ then $G/a \sim G/b$.
Proof: See [CR].

Lemma 2 let G and H be an in-trees, $a_1, a_2 \in I(G)$, and $b_1, b_2 \in I(H)$.
If

$a_1 \neq a_2$ and $\lambda(a_1) \geq \lambda(a_2) \geq \lambda(a)$ for all $a \in I(G) - \{a_1, a_2\}$ and

$b_1 \neq b_2$ and $\lambda(b_1) \geqslant \lambda(b_2) \geqslant \lambda(b)$ for all $b \in I(H) - \{b_1, b_2\}$

then

$G \lesssim H$ implies $G/a_1 \lesssim H/b_1$ and $G/a_2 \lesssim H/b_2$,

$G < H$ implies $G/a_1 < H/b_1$ or $G/a_2 < H/b_2$ or both,

$G \sim H$ implies $G/a_1 \sim H/b_1$ and $G/a_2 \sim H/b_2$.

Proof: See [CR].

The next theorem relates the values of the solution to the optimality equations for the makespan to the relations on in-trees defined above.

Theorem A Let G and H be in-trees. Then

a) $G \sim H$ implies $M_G = M_H$,

b) $G \lesssim H$ implies $M_G \geqslant M_H$,

c) $G < H$ implies $M_G > M_H$.

Proof: The proof is by induction on $|V(H)|$, the number of tasks in H. Notice that in all three cases a, b, and c that $|V(G)| \leqslant |V(H)|$.

Basis: $|V(H)| = 0$.

The theorem is trivially true in this case.

Induction Step: Assume that $|V(H)| > 0$ and that the theorem is true for in-trees with fewer tasks than H.

Part a) We must show that $G \sim H$ implies $M_G = M_H$.

We break this part of the proof down into two cases, one in which H has exactly one initial task and one in which H has two or more initial tasks.

Case 1: H has exactly one initial task, say a.

By hypothesis and the properties of $\sim$ and in-trees, G is isomorphic to H. Therefore $M_G = M_H$.

Case 2: H has two or more initial tasks.

By the definition of $\sim$, G also has two or more initial tasks. By the definition of M_G we have

$$M_G = \frac{1}{2} \max_{\substack{a \neq b \\ a, b \in I(G)}} (L_2 M_{G/a} + L_2 M_{G/b}).$$

We have from Lemma 1 that if $\lambda(a) \geqslant \lambda(b)$ then $G/a \lesssim G/b$. By induction we have $M_{G/a} \geqslant M_{G/b}$ (We have invoked part b of the theorem with G/a playing the role of G and G/b playing the role of H.). Since is L_2 is monotone, $(L_2 M_{G/a} + L_2 M_{G/b})$ is maximized by choosing a and b with the highest possible levels. Therefore,

$$M_G = \frac{1}{2} (L_2 M_{G/a_1} + L_2 M_{G/a_2}). \tag{1}$$

where $a_1, a_2 \in I(G)$, $a_1 \neq a_2$ and $\lambda(a_1) \geqslant \lambda(a_2) \geqslant \lambda(a)$ for all $a \in I(G) - \{a_1, a_2\}$. It should be clear that the choice of a_1 and a_2 is not necessarily unique since any other initial tasks at the same levels would serve.

Similarly, we have

$$M_H = \frac{1}{2} (L_2 M_{H/b_1} + L_2 M_{H/b_2}). \tag{2}$$

where $b_1, b_2 \in I(H)$, $b_2 \neq b_2$ and $\lambda(b_1) \geqslant \lambda(b_2) \geqslant \lambda(b)$ for all $b \in I(G) - \{b_1, b_2\}$.

By lemma 2 we have that $G \sim H$ implies that $G/a_1 \sim H/b_1$ and $G/a_2 \sim H/b_2$. It follows from the induction hypothesis that

$$M_{G/a_1} = M_{H/b_1}$$

and

$$M_{G/a_2} = M_{H/b_2}.$$

Combining these results with equations (1) and (2) we get that $M_G = M_H$, the desired result.

Part b) We must show that $G \leqslant H$ implies $M_G \geqslant M_H$.

We break this part of the proof down into 4 cases depending on whether the in-trees have one or more than one initial task.

Case 1: H and G both have at least two initial tasks.

It is easy to see that equations (1) and (2) hold in this case. By lemma 2 we have that $G \leqslant H$ implies that $G/a_1 \leqslant H/b_1$ and $G/a_2 \leqslant H/b_2$. It follows from the induction hypothesis that

$$M_{G/a_1} \geqslant M_{H/b_1}$$

and

$$M_{G/a_2} \geqslant M_{H/b_2}.$$

Combining this with equations (1) and (2) and by using the monotonicity of the operator L_2, we get $M_G \geqslant M_H$.

Case 2: H has one initial task and G has at least two initial tasks.

We use a sample-path analysis for this case. Clearly, $|V(G)| \leqslant |V(H)|$ and H is a path. Consider a policy for G which assigns the two processors to $a, b \in I(G)$ at $t = 0$ and after the first finish uses only one of the processors thereafter. Let Y denote the latest completion time of the tasks in G under this policy. Also, let τ_u denote the processing time of task u. We can write the random variable Y as follows,

$$Y = \sum_{u \in V(G)} \tau_u - min(\tau_a, \tau_b).$$

Since H has one initial task, we will only have use for one processor. Let Z be the latest completion time of the tasks in H. We can write Z as follows,

$$Z = \sum_{v \in V(H)} \tau_v .$$

Let ϕ be a 1-1 mapping of the tasks of G onto a subset of the tasks of H. Using ϕ we have,

$$Z = \sum_{u \in V(G)} \tau_{\phi(u)} + \Psi$$

where $\Psi \geqslant 0$. Combining expressions for Y and Z we get

$$Y = Z - (min(\tau_a, \tau_b) + \Psi). \tag{3}$$

Clearly $Pr\{Z \leqslant t\} = M_H(t)$. The optimality equations for M_G determine an optimal policy so clearly $Pr\{Y \leqslant t\} \leqslant M_G(t)$. Using (3) we get

$$M_G(t) \geqslant Pr\{Y \leqslant t\} > Pr\{Z \leqslant t\} = M_H(t).$$

Case 3: H has at least two initial tasks and G has one inital task.

There is only one policy for G. However H contains a path that is at least as long as G. Therefore, form a sample-path point of view, H will always take at least as long as G.

CASE 4: H and G both have one initial task.

Easy.

Part c) We must show that $G < H$ implies $M_G > M_H$.
The proof is similar to Part b).
End of Proof

We now are in a position to prove Theorem 1 of section 5.
Proof of Theorem 1: To show that a HLF policy π is makespan optimal we must show that $M_G^\pi = M_G$ for all in-trees G. In addition, if we want to show that only the HLF policies are makespan optimal, then we will also have to show that if π is not HLF then there exists an in-tree G such that $M_G^\pi < M_G$.
The proof is by induction on $|V(G)|$ for the first part, namely that if π is HLF then π is a makespan optimal policy.
Basis : $|V(G)| = 0$. The policy π is clearly optimal.
Induction Step : $|V(G)| > 0$ and we assume that π is makespan optimal for all in-trees with fewer tasks than G. If G has only one initial task then all policies are identical. Therefore we can assume that G has at least two initial tasks. Suppose $\pi(G) = \{u, v\}$. Then we can write

$$M_G^\pi = \frac{1}{2}(L_2 M_{G/u}^\pi + L_2 M_{G/v}^\pi).$$

By induction we get

$$M_G^\pi = \frac{1}{2}(L_2 M_{G/u} + L_2 M_{G/v}). \tag{4}$$

From Theorem A we can write(See equation (1)),

$$M_G = \frac{1}{2}(L_2 M_{G/a} + L_2 M_{G/b}) \tag{5}$$

where $a \neq b$ and $\lambda(a) \geqslant \lambda(b) \geqslant \lambda(c)$ for all $c \in I(G) - \{a, b\}$.
Clearly the choice of a and b is HLF but not necessarily the same as u and v. However, if $\lambda(u) \geqslant \lambda(v)$ then $\lambda(a) = \lambda(u)$ and $\lambda(b) = \lambda(v)$. By lemma 1 we have $G/a \sim G/u$ and $G/b \sim G/v$. Therefore using Theorem A part a we have that $M_{G/a} = M_{G/u}$ and $M_{G/b} = M_{G/v}$. Consequently, $M_G = M_G^\pi$.
To complete the proof let π be a nonHLF policy. Let G be the smallest in-tree such that $\pi(G)$ is not an HLF assignment. Therefore if $\pi(G) = \{u, v\}$, then there exists a $c \in I(G) - \{u, v\}$ such that either $\lambda(c) > \lambda(u)$ or $\lambda(c) > \lambda(v)$. Equation (4) still holds since G has been chosen to be as small as possible and equation (5) holds. Assume $\lambda(c) > \lambda(u)$. Then we can replace the term $M_{G/u}$ in equation (4) with the term $M_{G/c}$ making the right-hand-side(rhs) of (4) *strictly* larger. This follows from lemma 1 ($\lambda(c) > \lambda(u)$ implies $G/c < G/u$) and Theorem A, part c ($G/c < G/u$ implies $M_{G/c} > M_{G/u}$). We continue replacing terms on the rhs of (4) until it matches the rhs of (5). This process can be carried out without ever decreasing the rhs of (4). From this we conclude that $M_G^\pi < M_G$.
End of Proof

The proof of Theorem 2 follows the same pattern as the proof of Theorem 1 and consequently will be omitted.